Faculty of Computers and Information Technology

Department of Computer Science

AI-Powered Synthetic Data Generation

Supervisor's name

Dr. Yasser Kamal

Team Members

Habiba Khalil

Abdelrahman Hussein

Mohamed Wael

Mohamed Ebrahim

Amr Sayed

Academic Year: 2025–2026

Abstract

This project presents an AI-powered synthetic data generation system designed to produce high-quality, privacy-preserving, and customizable tabular datasets for analytical and machine learning applications. The solution aims to simplify data preparation, eliminate privacy concerns, and accelerate model development by enabling users to generate realistic synthetic datasets through an intuitive web interface. The platform adaptively selects the most suitable data generation approach based on user requirements, making it accessible to both technical and non-technical users. The outcome is a scalable, user-friendly tool that ensures data utility, diversity, and confidentiality across different domains.

Acknowledgements

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
DM	Data Mining

1 Introduction

This chapter introduces the project, gives background context, defines the problem, and outlines objectives and significance.

1.1 Background of the Study

Today, data is one of the most important assets for organizations, it is the source of decision-making, analytical systems, and machine learning models. The more industries depend on data-driven solutions, the more demand there will be for large, diverse, and high-quality datasets. However, acquiring real-world data is not an easy task, and this is because of the strict privacy regulations, limited availability, collection costs, and ethical issues. A number of institutions encounter problems like the sensitive nature of the data, incomplete datasets, or obstacles that hinder the sharing of data among teams or organizations.

Synthetic data generation has come up as a good option to these obstacles. Generative models and artificial intelligence allow creating datasets that are realistic, statistically accurate, and at the same time, privacy-preserving. These datasets are characterized by real data but do not reveal sensitive information. Techniques like Generative Adversarial Networks (GANs), especially CTGAN, have been able to learn very well the patterns from tabular data. Moreover, rule-based data simulators such as Mimesis can deliver personalized datasets in the absence of original data.

In spite of these developments, a major difference is still present between powerful AI methods and easy-to-use tools that let non-experts create synthetic data with no difficulty. The majority of current solutions are either too complicated, overly restricted in functionality, or included only as part of expensive enterprise products. This condition is calling for a versatile and user-friendly platform that permits users to create synthetic data according to their specific needs, whether they supply an existing dataset or ask for a completely new artificial dataset created from scratch.

This project addresses that need by developing an AI-powered system that automates synthetic data generation through a simple, intuitive interface, reducing technical barriers and enabling broader adoption across academic, industrial, and research fields.

1.2 Problem Statement

The major limitations of current data-driven applications are mainly attributed to privacy policies, unavailability of datasets, and the expensive nature of collecting real-world data. The available tools for generating synthetic data, on the other hand, are mostly too complicated for non-experts, too rigid, or they expect profound technical skills for their use. Thus, it is a common case that organizations do not get the desired data that is neither realistic nor safe from prying eyes. The present research intends to bridge this divide by creating a friendly AI system that can produce synthetic tabular data as per the user requirement .The promised outcome is an approachable solution that keeps data usage, confidentiality, and comfort as the main priorities.

1.3 Aim and Objectives

General Aim

The project intends to create an AI-based system that can produce synthetic tabular data that is of excellent quality, realistic and at the same time, respect privacy using powerful generative models along with an easy-to-use web interface thus allowing both technical and non-technical users to generate personalized datasets according to their particular needs.

Specific Objectives

In working towards the main aim, the project will concentrate on these objectives:

1- Create a synthetic data generation machine for data-driven generation and rule-based generation when no dataset is available.

2- Create and build a user-friendly and engaging web interface that enables users to state their needs, upload datasets and synthesize data with little technical know-how.

3- Control data quality and usefulness by checking the synthetic data produced against real data using statistical and machine learning performance metrics.

4- Add privacy-preserving features that will make sure the synthetic data will not give away any sensitive or identifiable information.

5- Allow for the scaling up and flexible customization so that users can specify dataset size, data types, distributions, and generation mode according to their requirements.

6- Provide a effortless workflow that automates the entire process from requirement input to generation ensuring a smooth experience.

1.4 Scope of the Project

This project is aimed at the development of an AI-powered system that can generate synthetic tabular data, designed and implemented in two ways:
(1) **Data-driven generation** using CTGAN technique when the user uploads a real dataset, and
(2) **Rule-based generation** Mimesis when no sample data is provided by the users.

The project includes developing a user-friendly web interface, where users will be able to specify their dataset requirements, select the generation modes, change the parameters like size and data types, and also download the final synthetic dataset.

However, the project excludes image, audio, video, and time-series synthetic data generation.

hardware limitations associated with training GAN models on standard computing resources.

1.5 Significance of the Project

This project provides an easy and safe way to generate synthetic tabular data without exposing real sensitive information. It helps **students, researchers, and developers** who need realistic datasets for testing, training models, or running experiments when real data is unavailable, not enough or restricted.

Organizations benefit from the ability to share data internally or externally without privacy risks. The system also reduces time and effort by allowing users to generate data through a simple interface rather than dealing with complex models.

Overall, the project makes synthetic data generation more accessible, supports secure data handling, and improves the availability of quality datasets for various applications.

1.6 Project Methodology (Brief Overview)

This project follows a structured development approach inspired by the **Software Development Life Cycle (SDLC)** with iterative elements similar to Agile methodology. The development process includes gathering user requirements, designing system architecture, integrating AI models, building the user interface, testing functionality, and final deployment.

The backend synthetic data generation engine will be implemented in **Python**, utilizing **CTGAN** for data-driven generation and **Mimesis** for rule-based dataset creation. These models will operate as standalone modules that communicate with the web system via APIs.

The web platform will be developed using a **.NET framework for the backend services** and **Node.js** for handling front-end interactions, routing, and API requests. The interface will provide users with options to upload datasets, configure required parameters, and generate synthetic data through clear and interactive components.

Key technologies planned for the project include:

- **Python** (CTGAN, Mimesis, pandas, PyTorch)

- **.NET** for backend logic, API endpoints, and server-side operations

- **Node.js** for client-side integration and real-time communication

- **RESTful APIs** for connecting the AI model with the web platform

- **SQL databases** for storing configurations or logs if needed

This methodology ensures a modular, scalable, and user-friendly system where the AI engine and web interface operate smoothly together presenting a dependable synthetic data generation process.

1.7 Project Organization

This report is organized into several chapters that present the project in a clear and structured manner.
Chapter One introduces the project by outlining its background, problem statement, objectives, scope, significance, methodology, and the overall organization of the report.
Chapter Two presents a detailed literature review, covering existing synthetic data generation techniques, related studies, current challenges, and the technologies that form the foundation of modern synthetic data systems.
Chapter Three provides the system analysis and design, including the proposed concept, functional and non-functional requirements, system architecture, diagrams, and all relevant design artifacts.
Chapter Four will cover system implementation and integration details, while Chapter Five will focus on testing, evaluation, and performance analysis of the generated synthetic data.

2 Literature Review

2.1 Introduction

This chapter presents an overview of current research, tools, and technologies related to synthetic data generation. It examines existing models, platforms, and methodologies used for producing artificial tabular data, with a particular focus on generative models such as GAN-based architectures and rule-based data simulation techniques.
The aim of this chapter is to establish a foundation for the proposed solution by highlighting what has already been achieved in this domain, identifying the strengths and limitations of existing systems, and clarifying the motivation for developing a new AI-powered synthetic data generation platform.

2.2 Review of Related Systems / Studies

Over the past years, synthetic data generation has gained significant attention due to privacy concerns, data scarcity, and the need for diverse datasets in machine learning applications. Several tools and studies have contributed to this field:

GAN-Based Synthetic Data Generators

- **CTGAN (Conditional Tabular GAN):**
  A widely used generative model specifically designed for tabular data. CTGAN handles imbalanced columns, mixed data types, and complex dependencies between variables. Studies show that CTGAN produces highly realistic datasets that retain statistical properties similar to real data. However, it requires sufficient training data and computing resources, making it less suitable for users without technical expertise.

- **TVAE (Tabular Variational Autoencoder):**
  Another generative model for tabular data. It performs well with numeric data but may struggle with categorical variables compared to CTGAN.

Rule-Based Data Generation Tools

- **Mimesis:**
  A Python library that generates structured data based on predefined rules. It supports generating personal information, addresses, dates, numeric fields, and more. While highly customizable, it does not learn patterns from real datasets.

- **Faker:**
  A popular library used to generate fake names, emails, and demographic data. It is simple but limited in customization and not suitable for generating domain-specific tabular datasets.

  open-source platforms

- **Gretel.ai:**
  Provides high-quality synthetic data generation using advanced generative models. It offers privacy assurances and powerful features but is subscription-based and may be too complex for basic users.

- **MostlyAI**:
  A user-friendly synthetic data platform with strong privacy features and high-fidelity data generation. However, it is part of enterprise-level solutions that may not suit academic or small project use cases.

Academic Insights and Challenges

Research consistently emphasizes the need for synthetic data systems that balance privacy, realism, and usability. Key insights from existing studies include:

GAN-based models can unintentionally memorize sensitive data if not properly trained.

- Rule-based systems cannot capture the statistical dependencies present in real datasets.

- Many AI models require technical expertise, creating significant barriers for non-expert users.

- There remains a gap between high-performance models and easy-to-use tools accessible to a broader audience.

These findings support the motivation for developing a system that integrates both generative and rule-based approaches while prioritizing accessibility, privacy, and ease of use.

2.3 Research Gap / Summary of Literature Review

A various generative models and rule-based tools exist, most current solutions face several limitations.

Limited Dual-Mode Generation:
Few platforms support both data-driven generation (learning from real datasets) and rule-based generation (creating data from scratch).

Lack of Fully Open and Free Solutions:
High-quality commercial tools often require paid subscriptions, restricting access for students, researchers, and small organizations.

Privacy Validation Not Standardized:
Several studies point out that models may unintentionally memorize sensitive data, and many tools lack built-in privacy evaluation metrics.

Integration Challenges:
Most AI models are standalone scripts, not part of integrated web platforms that allow seamless configuration, execution, and download of synthetic datasets.

Addressing the Gap:
This project fills these gaps by developing a unified, user-friendly system that combines GAN-based machine learning generation with rule-based generation. The system prioritizes accessibility, customization, privacy, and ease of use through an intuitive web interface that caters to both technical and non-technical users.

2.4 Summary of Existing Technologies

Python Synthetic Data Toolkit

Python provides a flexible ecosystem for developing machine learning and data-generation systems. In this project, Python serves as the core environment for implementing the synthetic data engine. The **Synthetic Data Vault (SDV)** framework plays a central role, offering a comprehensive suite of tools and models specifically designed for generating high-quality tabular synthetic data. SDV provides unified interfaces, evaluators, and utilities that streamline model training, performance validation, and dataset synthesis.

Libraries such as **CTGAN**, implemented within the SDV ecosystem and built on top of **PyTorch**, deliver advanced generative modeling capabilities by learning complex patterns and relationships from real datasets. The **pandas** library supports efficient data manipulation, preprocessing, and tabular transformations, enabling the system to seamlessly prepare datasets for model training.

In addition to AI-driven methods, **Mimesis** provides a rule-based generation approach, allowing users to create fully synthetic datasets without relying on real data samples. This is particularly useful when custom schemas or predefined attribute types are required.

Together, SDV, CTGAN, PyTorch, pandas, and Mimesis form a robust and versatile Python-based toolkit capable of generating accurate, diverse, and privacy-preserving synthetic datasets while supporting both data-driven and rule-based generation modes.

.NET Backend Framework

The .NET framework serves as the primary backend technology for managing application logic, hosting API endpoints, and coordinating communication between the web interface and the Python-based synthetic data engine. .NET provides a stable, high-performance runtime environment with strong support for scalability, security, and modular development.
Through its structured architecture, .NET efficiently handles user requests, validates parameters, manages background processes, and handles the transfer of generated datasets. Its powerful middleware pipeline and support for asynchronous processing make it well-suited for building enterprise-grade services with consistent performance and reliability.

Node.js Service Layer

Node.js is utilized as an integration layer responsible for managing client-side interactions, routing requests, and facilitating communication between the front-end environment and backend services. As a fast, event-driven, non-blocking JavaScript runtime, Node.js is ideal for handling real-time operations and asynchronous communication with the .NET backend and Python engine.
This layer ensures seamless user interactions by processing configuration inputs, forwarding dataset generation requests, and handling result delivery. Node.js also manages API communication, user interface responsiveness, and error handling, thereby enhancing overall system efficiency.

RESTful APIs

RESTful APIs provide a standardized and stateless architectural style for enabling communication between different components of a software system. They are built on top of the HTTP protocol and define a set of constraints that promote scalability, simplicity, and reliability.
In this project, RESTful APIs serve as the communication backbone between the front-end interface, the .NET backend, and the Python-based synthetic data generation engine. By adhering to REST principles, the system ensures that each operation is clearly structured and accessible through well-defined endpoints.

- **POST** – Used for submitting data, such as uploading a dataset or sending generation parameters to the backend.
- **GET** – Used to retrieve generated datasets, system status, or configuration information.
- **PUT** – Used for updating existing resources or modifying system configurations.
- **PATCH** – Used to apply partial updates to specific fields of a resource.
- **DELETE** – Used to remove stored configurations, logs, or temporary data.

By utilizing RESTful APIs, the system achieves loose coupling between components, allowing each part (front-end, backend, Python engine) to evolve independently. This architecture improves scalability, simplifies debugging, and provides flexibility for integrating additional features or external systems.

SQL Database Systems

SQL databases are used to store system configurations, logs, user settings, and temporary metadata required for managing synthetic data generation workflows. SQL (Structured Query Language) enables reliable data retrieval, insertion, updating, and deletion through a standardized syntax.
Its ability to enforce constraints, maintain relationships, and ensure data consistency makes it essential for a project that requires accuracy and traceability. SQL databases also support indexing and optimized queries, contributing to system performance when handling large datasets or multiple user sessions.

HTML, CSS, and JavaScript

HTML, CSS, and JavaScript form the core technologies behind the web interface. HTML provides the structural layout for pages, enabling the system to define input forms, upload components, tables, and interactive controls.
CSS is used for designing the presentation layer, controlling themes, responsive layouts, and improving user experience through clean and modern styling.
JavaScript enables real-time interaction, client-side validations, dynamic page updates, and seamless communication with backend APIs. Together, these technologies allow the system to deliver an intuitive, interactive, and user-friendly interface that accommodates both technical and non-technical users.

3 System Analysis and Design

3.1 System Overview / Proposed Concept

The proposed system is an AI-powered synthetic data generation platform designed to produce realistic, customizable, and privacy-preserving tabular datasets. It integrates both **data-driven generative models** and **rule-based simulation techniques** to accommodate a wide range of user requirements. The system is developed with a modular architecture that separates the web interface, backend services, and AI generation engine, ensuring scalability, maintainability, and clear component interaction.

At its core, the platform provides two operational modes:

1. **Data-Driven Generation**
   In this mode, users upload a real dataset, and the system utilizes CTGAN models implemented within the SDV framework to learn statistical patterns, distributions, and correlations. The model then generates synthetic data that reflects the properties of the original dataset while preventing exposure of sensitive or identifiable information.

2. **Rule-Based Generation**
   If no real dataset is provided, the system uses Mimesis to generate synthetic data entirely from predefined rules and attribute types. Users specify attributes such as names, numbers, categories, and formats, allowing the system to construct structured datasets tailored to custom requirements.

The platform is exposed through an intuitive web interface where users can upload datasets, configure generation parameters, select the desired mode, and download the resulting synthetic data. The **.NET backend** handles request processing and system logic, while **RESTful APIs** facilitate communication between the web interface and the Python-based synthetic data engine. The **Node.js layer** manages client-side interactions and provides seamless integration between the user interface and backend operations.

By combining advanced AI models with flexible rule-based generation, the system provides a comprehensive solution suitable for academic research, machine learning model development, data analysis, and secure data sharing. The result is a user-friendly and reliable platform that reduces the technical barriers typically associated with synthetic data generation and enables users to obtain high-quality datasets efficiently and safely.

3.2 Functional Requirements

3.2.1 All Users

3.2.1.1. **View Platform Information:** All users shall be able to view general information about the system and its purpose.

3.2.1.2. **Sign Up:** All users shall be able to register with a username, email, and password.

3.2.1.3. **Login:** All users shall be able to log in with their registered credentials.

3.2.1.4. **Reset Password:** All users shall be able to reset their password if they forget it.

3.2.1.5. **Edit Profile:** All users shall be able to update their profile information.

3.2.1.6. **Delete Profile:** All users (except Admin) shall be able to delete their accounts.

3.2.2 Registered User

3.2.2.1. **Upload Dataset:** Registered users shall be able to upload datasets for data-driven synthetic generation.

3.2.2.2. **Validate Dataset:** Registered users shall be able to validate dataset structure before processing.

3.2.2.3. **Select Generation Mode:** Registered users shall be able to choose between CTGAN (data-driven) and Mimesis (rule-based) generation.

3.2.2.4. **Configure Generation Parameters:** Registered users shall be able to set the number of records, data types, and output formats.

3.2.2.5. **Generate Synthetic Data:** Registered users shall be able to initiate synthetic data generation.

3.2.2.6. **Preview Output:** Registered users shall be able to preview a sample of the generated dataset.

3.2.2.7. **Download Results:** Registered users shall be able to download the full synthetic dataset in supported formats (CSV, Excel).

3.2.2.8. **Use Rule-Based Generator:** Registered users shall be able to define custom schemas for Mimesis-based synthetic generation.

3.2.2.9. **View Generation History:** Registered users shall be able to view previously generated datasets and their configurations.

3.2.2.10. **Delete Generation History:** Registered users shall be able to delete previous generation logs or stored temporary files.

3.2.2.11. **View Real-Time Generating:** Registered users shall be able to receive status updates for long-running generation tasks.

3.2.3 Admin

3.2.3.1. **Manage User Accounts:** Admin shall be able to view, edit, block, or delete any user account.

3.2.3.2. **View Activity Logs:** Admin shall be able to view all system logs, including dataset uploads, generation tasks, and errors.

3.2.3.3. **Manage Data Storage:** Admin shall be able to clear temporary files, logs, and stored datasets.

3.2.3.4. **View Error Reports:** Admin shall be able to view system errors, failed jobs, and processing issues.

3.3 Non-Functional Requirements

3.3.1 Performance Requirements

3.3.1.1. The system **shall respond to user actions within 5 seconds** for standard operations such as navigation and parameter configuration.

3.3.1.2. The system **shall process dataset uploads up to a predefined size limit** (e.g., 50MB) without performance degradation.

3.3.1.3. The system **shall generate small to medium synthetic datasets efficiently**, returning results within acceptable time frames depending on dataset size and model complexity.

3.3.1.4. The system **shall support concurrent user requests** without affecting processing speed or causing noticeable delays.

3.3.2 Usability Requirements

3.3.2.1. The system **shall provide a clean, intuitive, and simple user interface** suitable for both technical and non-technical users.

3.3.2.2. The system **shall provide guidance text or documentation** to guide users through dataset upload, parameter selection, and generation processes.

3.3.2.3. The interface **shall support responsive design**, ensuring full usability across desktops, laptops and different operating systems.

3.3.2.4. The system **shall display user-friendly error messages** that explain issues clearly without exposing technical details.

3.3.2.5. The system **shall allow users to complete core workflows** (upload → configure → generate → download) with minimal steps.

3.3.3 Reliability Requirements

3.3.3.1. The system **shall maintain stable operation** under normal and peak loads.

3.3.3.2. The system **shall handle unexpected failures gracefully**, without losing user data or crashing the application.

3.3.3.3. The system **shall validate input data** to prevent errors in the processing pipeline.

3.3.4 Security Requirements

3.3.4.1. The system **shall encrypt user credentials** and sensitive information using industry-standard methods.

3.3.4.2. The system **shall ensure that uploaded datasets are not stored permanently**, and temporary data shall be removed after processing.

3.3.4.3. The system **shall restrict unauthorized access** to backend services, APIs, and internal data pipelines.

3.3.5 Scalability Requirements

3.3.5.1. The system **shall support horizontal scaling** of backend services to accommodate increased user load.

3.3.5.2. The architecture **shall allow integration of additional generation models** or new rule-based libraries without redesigning the system.

3.3.5.3. The system **shall support future increases in dataset size limits** as storage and computing resources grow.

3.3.5.4. The system **shall maintain performance levels** as the number of registered users or generated datasets increases.

3.4 System Environment

3.4.1 Hardware Environment

- Development Machines
  A standard computer or laptop with a minimum of:
  Processor: Intel Core i5 / AMD Ryzen 5 or higher
  RAM: 8 GB minimum
  Storage: At least 5 GB free space for datasets, logs, and model dependencies
  Operating System: Windows 10/11, Linux, or macOS

2. Server / Deployment Environment

Cloud or on-premise server

3.4.2 Software Environment

1. Backend Software

- **.NET Framework / .NET Core**
  Used for backend services, business logic, API endpoints, and handling user requests.

- **Node.js**
  Used as an integration layer to manage client-side interactions and communication with backend services.

2. AI & Data Generation Environment

- **Python 3.x**
  Base programming environment for implementing the synthetic data engine.

- **SDV (Synthetic Data Vault)**
  Framework for generative models.

- **CTGAN**
  Model used for data-driven tabular synthetic data generation.

- **PyTorch**
  Deep learning framework used for model training and computation.

- **pandas**
  Library for dataset preprocessing, validation, and manipulation.

- **Mimesis**
  Rule-based synthetic data generation library for schema-based datasets.

3. Database Environment

- **SQL Database**
  Used for storing system configurations, logs, user profiles, and optional metadata.

4. Frontend Software

- **HTML5**
  Used for structuring the user interface components.

- **CSS3**
  Used for styling, layout control, and responsive design.

- **JavaScript**
  Used for client-side logic, dynamic updates, and interaction with RESTful APIs.

5. API & Communication Environment

- **RESTful APIs**
  Used for communication between the frontend, .NET backend, and Python engine.

- **JSON**
  Used as the data format for exchanging requests and responses.

6. Development Tools

- **Visual Studio / Visual Studio Code**
  Used for coding, debugging, and project management.

- **Postman**
  Used for testing RESTful API endpoints during development.

- **Git / GitHub**
  Used for version control and collaborative development.

3.4.3 Operating Environment

The system will operate through a web browser, requiring no installation for end-users.

Supported browsers include:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Safari

The backend and AI engine run on a server environment, while users access the system via the web interface.

3.5 System Architecture

Place architectural diagrams here.

3.6 Class Diagram

3.7 Activity Diagram (Optional)

3.8 Use Case Diagram

3.9 Sequence Diagram

3.10 ERD Diagram

Bibliography